Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Benders Decomposition for Large-Scale Uncapacitated Hub Location

Ivan Contreras, Jean-François Cordeau, Gilbert Laporte,

To cite this article:
Ivan Contreras, Jean-François Cordeau, Gilbert Laporte, (2011) Benders Decomposition for Large-Scale Uncapacitated Hub Location. Operations Research 59(6):1477-1490. http://dx.doi.org/10.1287/opre.1110.0965

OPERATIONS RESEARCH
Vol. 59, No. 6, November–December 2011, pp. 1477–1490
ISSN 0030-364X | EISSN 1526-5463 | 11 | 5906 | 1477

http://dx.doi.org/10.1287/opre.1110.0965

Benders Decomposition for Large-Scale Uncapacitated Hub Location

Ivan Contreras
Concordia University and CIRRELT, Montréal H3G 1M8, Canada, ivan.contreras@cirrelt.ca

Jean-François Cordeau, Gilbert Laporte
HEC Montréal and CIRRELT, Montréal H3T 2A7, Canada
{jean-francois.cordeau@hec.ca, gilbert.laporte@cirrelt.ca}

This paper describes an exact algorithm capable of solving large-scale instances of the well-known *uncapacitated hub location problem with multiple assignments*. The algorithm applies Benders decomposition to a strong path-based formulation of the problem. The standard decomposition algorithm is enhanced through the inclusion of several features such as the use of a multicut reformulation, the generation of strong optimality cuts, the integration of reduction tests, and the execution of a heuristic procedure. Extensive computational experiments were performed to evaluate the efficiency and robustness of the algorithm. Computational results obtained on classical benchmark instances (with up to 200 nodes) and on a new and more difficult set of instances (with up to 500 nodes) confirm the efficiency of the algorithm.

Subject classifications: hub location; Benders decomposition, Pareto-optimal cuts; elimination tests.
Area of review: Transportation.
History: Received May 2010; revisions received September 2010, November 2010; accepted February 2011.

1. Introduction

Transportation, telecommunications, and computer networks frequently employ hub-and-spoke architectures to efficiently route commodities between many origins and destinations. Their key feature lies in the use of consolidation, switching, or transshipment points called *hub facilities* to connect a large number of origin/destination (O/D) pairs by using a small number of links. This helps reduce setup costs, centralize commodity handling and sorting operations, and achieve economies of scale on routing costs through the consolidation of flows.

Hub location problems (HLPs) constitute a difficult class of $\mathcal{NP}$-hard combinatorial optimization problems combining location and network design decisions. Their main difficulty stems from the inherent interrelation between two levels of the decision process. The first level considers the selection of a set of nodes to locate hub facilities, whereas the second level deals with the design of the hub network, usually determined by the allocation pattern of nodes to hub facilities.

The field of hub location is rooted in the work of O'Kelly (1986) and has since evolved into a rich research area. We refer the reader to some of the main survey articles on this topic. The early reviews dealing with HLPs, by O'Kelly and Miller (1994) and Campbell (1994a), contain classification schemes for the existing models and for the topological structures applicable to hub networks. Klincewicz (1998) later presented a survey on the design of hub networks in the context of telecommunication networks, and Bryan and O'Kelly (1999) concentrated on air transportation networks. Campbell et al. (2002) wrote a comprehensive survey on network hub location problems in which the location of hubs is the key decision. A more recent paper, by Alumur and Kara (2008), provides an updated and extensive review of the growing literature on network hub location models.

Despite the considerable efforts already made by many researchers, the optimal solution of HLPs remains challenging, particularly when considering more realistic, large-scale instances. To give an idea of the inherent difficulty of HLPs, instances with more than 50 nodes cannot be solved optimally for the vast majority of the variants considered in the literature, and it is only very recently that for some limited classes of HLPs, instances with up to 200 nodes have been solved optimally (see Camargo et al. 2008, Contreras et al. 2011).

In this paper we present an exact algorithm capable of solving large-scale instances for one of the most classical problems in the hub location literature, the *Uncapacitated Hub Location Problem with Multiple Assignments* (UHLPMA). In this problem, commodities must be routed via a set of hubs, and thus paths between O/D pairs must include at least one hub facility. It is assumed that hubs are fully interconnected with more effective, higher-volume pathways, which allow a constant discount factor τ ($0 < \tau < 1$) to be applied to interhub transportation costs. The transportation costs between nodes are assumed to be symmetric and to satisfy the triangle inequality. The incoming

and outgoing flows at the hub facilities and the flow routed through each link of the network are unbounded. The number of hubs to locate is not known a priori, but a fixed set-up cost for each hub is considered. The objective is to minimize the sum of hub fixed costs and of demand transportation costs over the network. We consider the case in which multiple allocations are allowed, i.e., each O/D point may send and receive commodities through several hubs. Note that a multiple assignment pattern is crucial when minimizing the total transportation cost (Campbell 1996).

There exist several papers on the UHLPMA. The first mathematical programming model was introduced by Campbell (1994b), but was not computationally tested. Since then, several efforts have been made to produce better and tighter mixed-integer programming (MIP) formulations. Boland et al. (2004) have developed a multicommodity flow-based formulation capable of producing optimal solutions for instances with up to 50 nodes by using a general-purpose solver. Later, Hamacher et al. (2004) and Marín et al. (2006) presented path-based formulations yielding much tighter LP bounds. However, due to their size, these formulations were only able to optimally solve instances with up to 25 nodes using general-purpose solvers. Marín et al. (2006) also presented formulations for a more general case in which distances do not necessarily satisfy the triangle inequality. The first exact algorithms, put forward by Klincewicz (1996) and by Mayer and Wagner (2002), were branch-and-bound (BB) methods based on dual-ascent and dual-adjustments techniques. In particular, the HubLocator algorithm (Mayer and Wagner 2002) was able to obtain optimal solutions for instances with up to 40 nodes. Marín (2005) proposed a relax-and-cut algorithm that could solve to optimality instances with up to 50 nodes. Later, Cánovas et al. (2007) introduced a new BB method, also based on a dual-ascent strategy. This method was able to solve to optimality instances with up to 120 nodes. Recently, Camargo et al. (2008) presented an exact Benders decomposition algorithm that was applied to instances involving up to 200 nodes. To the best of our knowledge, these instances are the largest ever solved exactly for any type of uncapacitated hub location problem.

The main contribution of this paper is to propose an exact algorithm applicable to large-scale symmetric instances of the UHLPMA involving up to 500 nodes and 250,000 commodities. It is a Benders decomposition algorithm based on the path-based formulation of Hamacher et al. (2004). The basic implementation of the algorithm is enhanced through several algorithmic features that make it more robust and efficient. These include: (i) the use of a stronger multicut Benders reformulation, (ii) the generation of stronger cuts that approximate Pareto-optimal cuts, (iii) the inclusion of reduction tests during the inner iterations of the Benders decomposition algorithm and, (iv) the use of a heuristic for the a priori generation of optimality cuts. In order to evaluate and assess the robustness, efficiency, and limitations of our proposed algorithm, extensive computational experiments were performed on symmetric instances from the classical Australian Post data set and from a new challenging set of instances.

The remainder of the paper is organized as follows. Section 2 formally defines the problem and presents an MIP formulation as well as properties of optimal solutions. The basic Benders reformulation, the Benders decomposition algorithm, and some aspects of the dual problem are then presented in §3. Section 4 introduces several features that improve the convergence and efficiency of the algorithm. Section 5 presents the results of extensive computational experiments performed on a wide variety of instances. Conclusions follow in §6.

2. Problem Definition

Let $G = (N, A)$ be a complete digraph, where N is the set of nodes and A is the set of arcs. Let $H \subseteq N$ be the set of potential hub locations, and K represent the set of commodities whose origin and destination points belong to N. For each commodity $k \in K$, define W_k as the amount of commodity k to be routed from the origin $o(k) \in N$ to the destination $d(k) \in N$. For each node $i \in H$, f_i is the fixed set-up cost for locating a hub. The distances, or transportation costs $d_{ij} \geqslant 0$ between nodes i and j, are assumed to be symmetric and satisfy the triangle inequality. The UHLPMA consists of locating a set of hubs and of determining the routing of commodity flows through the hub nodes, with the objective of minimizing the total set-up and transportation cost.

Given that hub nodes are fully interconnected and distances satisfy the triangle inequality, every path between an origin and a destination node will contain at least one, and at most two, hubs. For this reason, paths between two nodes are of the form $(o(k), i, j, d(k))$, where $(i, j) \in H \times H$ is the ordered pair of hubs to which $o(k)$ and $d(k)$ are allocated, respectively. Therefore, the transportation cost of routing commodity k along the path $(o(k), i, j, d(k))$ is given by $\hat{F}_{ijk} = W_k(\chi d_{o(k)i} + \tau d_{ij} + \delta d_{jd(k)})$, where χ, τ, and δ represent the collection, transfer, and distribution costs along the path. To reflect economies of scale between hubs, we assume that $\tau < \chi$ and $\tau < \delta$. We define binary location variables z_i, $i \in H$, equal to 1 if and only if a hub is located at node i. We also introduce binary routing variables x_{ijk}, $k \in K$, and $(i, j) \in H \times H$, equal to 1 if and only if commodity k transits via a first hub node i and a second hub node j. Following Hamacher et al. (2004), the UHLPMA can be stated as follows:

$$\text{minimize} \quad \sum_{i \in H} f_i z_i + \sum_{i \in H} \sum_{j \in H} \sum_{k \in K} \hat{F}_{ijk} x_{ijk}, \tag{1}$$

$$\text{subject to} \quad \sum_{i \in H} \sum_{j \in H} x_{ijk} = 1 \quad \forall k \in K \tag{2}$$

$$\sum_{j \in H} x_{ijk} + \sum_{j \in H \setminus \{i\}} x_{jik} \leqslant z_i \quad \forall i \in H, \forall k \in K \tag{3}$$

$$x_{ijk} \geqslant 0 \quad \forall i, j \in H, \forall k \in K \tag{4}$$

$$z_i \in \{0, 1\} \quad \forall i \in H. \tag{5}$$

The first term of the objective function represents the total set-up cost of the hub facilities and the second term is the total transportation cost. Constraints (2) guarantee that there is a single path connecting the origin and destination nodes of every commodity. Constraints (3) prohibit commodities from being routed via a nonhub node. Finally, constraints (4) and (5) are the standard nonnegativity and integrality constraints. Given that there are no capacity constraints on the hub nodes, there is no need to explicitly state the integrality on the x_{ijk} variables because there always exists an optimal solution of (1)–(5) in which all x_{ijk} variables are integer.

2.1. Properties of Optimal Solutions and Preprocessing

Several properties and characteristics of optimal UHLPMA solutions are known and can be used to perform preprocessing. In this section, we unify and summarize the most relevant results and present them in the context of the path-based formulation. We define a hub edge as a *set* $e \in E$, where E is the set of subsets of H containing one or two hubs. We denote e as $\{e_1, e_2\}$ if $|e| = 2$ and as $\{e_1\}$ if $|e| = 1$, i.e., e is a loop. We can eliminate approximately half of the x_{ijk} variables associated with nonoptimal directions by simply using an *undirected* transportation cost for every hub edge (Hamacher et al. 2004). In any optimal UHLPMA solution, every commodity uses at most one direction of a hub edge, the one with lower transportation cost. The *undirected* transportation cost F_{ek} for each $e \in E$ and $k \in K$ is defined as $F_{ek} = \min\{\hat{F}_{ijk}, \hat{F}_{jik}\}$ if $e = \{i, j\}$, and $F_{ek} = \hat{F}_{iik}$ if $e = \{i\}$. Moreover, it can be shown that in any optimal UHLPMA solution, no commodity k will be routed through a hub edge e containing two different hubs whenever it is cheaper to route it through only one of them (Boland et al. 2004, Marín et al. 2006).

PROPERTY 1. *For every* $k \in K$ *and* $e \in E$, $|e| = 2$, *such that* $F_{ek} > \min\{F_{\{e_1\}k}, F_{\{e_2\}k}\}$, $x_{ek} = 0$ *in any optimal UHLPMA solution.*

We now consider the particular case of commodities k having the same origin and destination points; that is, $o(k) = d(k)$. One can observe that such commodities will never be routed through two hubs whenever the distances are symmetric, i.e., $d_{ij} = d_{ji}$ for each $(i, j) \in A$. Indeed, they will always be collected and distributed by their closest open-hub facility (Boland et al. 2004).

PROPERTY 2. *If* $d_{ij} = d_{ji}$ *for each* $(i, j) \in A$, *then for every* $e \in E$, *such that* $|e| = 2$ *and* $k \in K$ *such that* $o(k) = d(k)$, $x_{ek} = 0$ *in any optimal UHLPMA solution.*

The above properties lead to a more compact formulation with fewer variables, but with the same number of constraints. We define a set of candidate hub edges for each commodity $k \in K$ as

$$E_k = \begin{cases} \{e \in E\colon |e| = 1\} \cup \{e \in E\colon |e| = 2 \quad \text{and} \\ \quad (F_{ek} < \min\{F_{\{e_1\}k}, F_{\{e_2\}k}\})\}, & \text{if } o(k) \neq d(k), \\ \{e \in E\colon |e| = 1\}, & \text{otherwise.} \end{cases}$$

The UHLPMA can thus be restated as

$$\text{minimize} \quad \sum_{i \in H} f_i z_i + \sum_{k \in K} \sum_{e \in E_k} F_{ek} x_{ek}, \tag{6}$$

$$\text{subject to} \quad \sum_{e \in E_k} x_{ek} = 1 \quad \forall k \in K \tag{7}$$

$$\sum_{e \in E_k : i \in e} x_{ek} \leqslant z_i \quad \forall i \in H, \forall k \in K \tag{8}$$

$$x_{ek} \geqslant 0 \quad \forall k \in K, \forall e \in E_k \tag{9}$$

$$z_i \in \{0, 1\} \quad \forall i \in H. \tag{10}$$

Finally, we consider the special case of symmetric transportation costs. Transportation costs for the commodity paths are symmetric when the cost of path (o, i, j, d) is equal to the cost of path (d, j, i, o), where $o, d \in N$ are O/D nodes and $i, j \in H$ are hub nodes.

PROPERTY 3. *If* $\chi = \delta$, *then* $F_{ek_1} = F_{ek_2}$ *for each* $e \in E$ *and each pair of commodities* (k_1, k_2) *such that* $o(k_1) = d(k_2)$ *and* $d(k_1) = o(k_2)$.

When transportation costs are symmetric, we can further reduce the number of x_{ek} variables and constraints by considering as one commodity $\hat{k}$ the sum of two commodities k_1, k_2 having the opposite O/D pairs; i.e., $W_{\hat{k}} = W_{k_1} + W_{k_2}$, where $\hat{k} = (o(k_1), d(k_2))$, $o(k_1) = d(k_2)$, and $d(k_1) = o(k_2)$.

3. Benders Decomposition

Benders decomposition is a partitioning method applicable to mixed-integer programs (Benders 1962). It separates the original problem into two simpler ones: an integer *master problem* and a linear *subproblem*. In this section, we introduce a Benders reformulation of the UHLPMA based on the compact formulation (6)–(10). We then describe a basic Benders decomposition algorithm to solve the reformulation. Because of degeneracy in the primal subproblem, there may exist multiple dual solutions. We thus present an efficient procedure to select, among the set of optimal-dual solutions, an *appropriate* solution capable of generating a strong cut for the master problem.

3.1. Benders Reformulation

Let $Z = \mathbb{B}^{|H|}$ denote the set of binary vectors associated with the z_i variables. For any fixed vector $\hat{z} \in Z$, the *primal*

subproblem (PS) in the space of the x_{ek} variables is

$$\text{(PS)} \quad v(\hat{z}) = \text{minimize} \sum_{k\in K}\sum_{e\in E_k} F_{ek}x_{ek},$$

$$\text{subject to} \quad (7),\ (9)$$

$$\sum_{e\in E_k:\, i\in e} x_{ek} \leqslant \hat{z}_i \qquad \forall\, i\in H,\ \forall\, k\in K. \quad (11)$$

Let α_k and u_{ik} be the dual variables associated with constraints (7) and (11), respectively. The *dual subproblem* (DS), which is the dual of PS, can be stated as follows:

$$\text{(DS)} \quad \text{maximize} \sum_{k\in K}\alpha_k - \sum_{i\in H}\sum_{k\in K}\hat{z}_i u_{ik},$$

$$\text{subject to} \quad \alpha_k - u_{e_1k} - u_{e_2k} \leqslant F_{ek} \qquad \forall\, k\in K,\ \forall\, e\in E_k,\ |e|=2 \quad (12)$$

$$\alpha_k - u_{e_1k} \leqslant F_{ek} \qquad \forall\, k\in K,\ \forall\, e\in E_k,\ |e|=1 \quad (13)$$

$$u_{ik} \geqslant 0 \quad \forall\, i\in H,\ \forall\, k\in K. \quad (14)$$

Let D denote the set of feasible solutions of DS and let P_D be the set of extreme points of D. Observe that D is not modified when changing $\hat{z}$ and, because $F_{ek}\geqslant 0$ for each $k\in K$ and $e\in E_k$, the null vector 0 is always a solution to DS. Hence, because of strong duality, either the primal subproblem is feasible and bounded, or it is infeasible. We are thus interested in $\hat{z}$ vectors that give rise to primal subproblems of the former case. The following result establishes under which condition such vectors exist.

Proposition 1. *For any vector $\hat{z}\in Z$ such that $\sum_{i\in H}\hat{z}_i \geqslant 1$, the primal and dual subproblems are feasible and bounded.*

Proof. For any vector $\hat{z}$ such that $\sum_{i\in H}\hat{z}_i\geqslant 1$, there exists at least one possible path x_{ek} for every commodity $k\in K$, and thus the primal problem is feasible. Moreover, because the transportation costs F_{ek} are finite and because of constraints (7) and (11), any feasible solution of PS must be bounded. By strong duality, the dual subproblem is also feasible and bounded. □

It follows that the dual objective function value is equal to

$$\max_{(\alpha,u)\in P_D} \sum_{k\in K}\alpha_k - \sum_{i\in H}\sum_{k\in K}\hat{z}_i u_{ik}. \quad (15)$$

Introducing an extra variable η for the overall transportation cost, we can formulate the Benders master problem (MP) as follows:

$$\text{(MP)} \quad \text{minimize} \sum_{i\in H} f_i z_i + \eta,$$

$$\text{subject to} \quad \eta \geqslant \sum_{k\in K}\alpha_k - \sum_{i\in H}\sum_{k\in K} u_{ik}z_i \qquad \forall\,(\alpha,u)\in P_D \quad (16)$$

$$\sum_{i\in H} z_i \geqslant 1 \quad (17)$$

$$z_i\in\{0,1\} \quad \forall\, i\in H. \quad (18)$$

Observe that Benders feasibility cuts associated with the extreme rays of D are not necessary in the Benders reformulation because the feasibility of PS is ensured by constraints (17). We have thus transformed problem (6)–(10) into an equivalent MIP problem with $|H|$ binary variables and one continuous variable. Nevertheless, the above Benders reformulation contains an exponential number of constraints and must be tackled through an adequate cutting plane approach. Thus, we iteratively solve relaxed master problems containing a small subset of the constraints (16) associated with the extreme points of P_D, and we keep adding these as needed by solving dual subproblems until an optimal solution to the original problem is obtained.

3.2. Basic Benders Decomposition Algorithm

Let ub denote an upper bound on the optimal solution value and let t represent the current iteration number. Let P_D^t denote the restricted set of extreme points of D at iteration t, $\text{MP}(P_D^t)$ the relaxed master problem obtained by replacing P_D by P_D^t in MP, and $v(\text{MP}(P_D^t))$ its optimal solution value. Also, let z^t be an optimal solution vector of $\text{MP}(P_D^t)$, $\text{DS}(z^t)$ the dual subproblem for z^t, and $v(\text{DS}(z^t))$ its optimal solution value. A pseudocode of the basic Benders decomposition algorithm is provided in Algorithm 1.

Algorithm 1 (Benders decomposition)

```
ub ← ∞, t ← 0
P_D^t ← ∅
terminate ← false
while (terminate = false) do
  Solve MP(P_D^t) to obtain z^t
  if (v(MP(P_D^t)) = ub) then
    terminate ← true
  else
    Solve DS(z^t) to obtain (α^t, u^t) ∈ P_D
    P_D^{t+1} ← P_D^t ∪ {(α^t, u^t)}
    if (v(DS(z^t)) + Σ_{i∈H} f_i z_i^t < ub) then
      ub ← v(DS(z^t)) + Σ_{i∈H} f_i z_i^t
    end if
  end if
  t ← t + 1
end while
```

Whenever the problem defined by (6)–(10) is feasible, Algorithm 1 will yield an optimal solution. The computational efficiency of the above Benders decomposition algorithm depends mainly on: (i) the computational effort needed to solve $\text{MP}(P_D^t)$, (ii) the computational effort needed to solve $\text{DS}(z^t)$, and (iii) the number of iterations required to obtain an optimal solution. Next, we present a methodology for efficiently solving $\text{DS}(z^t)$ by exploiting the structure of the primal subproblem. In §4, we will present some techniques focusing on (ii) and (iii).

3.3. Solving the Subproblem

At any iteration t of Algorithm 1, we obtain an optimal solution vector z^t of MP(P_D^t). Let $H_1^t = \{i\colon z_i^t = 1\}$ be the set of open hubs and $H_0^t = \{i\colon z_i^t = 0\}$ be the set of closed hubs. Given that $z^t \in Z$, we can exploit the structure of the primal subproblem to obtain a vector of optimal-dual variables (α^t, u^t) more efficiently than by using an LP solver for the explicit solution of DS.

In uncapacitated hub location problems with multiple assignments, once the location of hubs is known, the allocation decisions become trivial. Ernst and Krishnamoorthy (1998) have shown that the allocation subproblem is equivalent to an all-pairs shortest-path problem in a modified network. However, this method cannot benefit from the reduction in the number of variables provided by the preprocessing phase, which allows PS to be reduced to the equivalent problem:

$$(\mathrm{PS}^t) \quad \text{minimize} \sum_{k\in K}\sum_{e\in E_k} F_{ek}x_{ek}$$

$$\text{subject to} \sum_{e\in E_k\cap(H_1^t\times H_1^t)} x_{ek} = 1 \quad \forall k\in K,$$

$$x_{ek} \geqslant 0 \quad \forall k\in K,\ \forall e\in E_k.$$

This problem can be separated into $|K|$ independent subproblems PS_k^t, one for each commodity $k \in K$. Each PS_k^t is a *semiassignment* problem that can be easily solved by choosing exactly one hub edge, denoted as $e(k)$, among those with minimum transportation cost route associated with open hubs. For a given k, a primal optimal solution of PS_k^t, denoted by x^t, can be obtained by setting exactly one x_{ek} variable equal to one and the rest to zero, i.e., $x_{e(k)k}^t = 1$ for one element $e(k) \in \arg\min\{F_{ek}\colon e \in E_k \cap (H_1^t \times H_1^t)\}$ and $x_{ek}^t = 0$, for $e \in E_k\backslash\{e(k)\}$. The optimal solution value of PS at z^t, denoted as $v(z^t)$, can thus be expressed as

$$v(z^t) = \sum_{k\in K} F_{e(k)k} = \sum_{k\in K} \min\{F_{ek}\colon e\in E_k\cap(H_1^t\times H_1^t)\}. \quad (19)$$

In order to obtain an associated optimality cut, we still need to produce an optimal-dual solution (α^t, u^t). We can use duality theory to recover a dual solution (α^t, u^t) from the primal-optimal solution x^t. In particular, the complementary slackness conditions are

$$u_{ik}^t\left(\sum_{e\in E_k:\, i\in e} x_{ek}^t - z_i^t\right) = 0, \quad \forall i\in H,\ k\in K, \quad (20)$$

$$x_{ek}^t(\alpha_k^t - u_{e_1k}^t - u_{e_2k}^t - F_{ek}) = 0, \quad \forall k\in K,\ e\in E_k,\ |e| = 2, \quad (21)$$

$$x_{ek}^t(\alpha_k^t - u_{ek}^t - F_{ek}) = 0, \quad \forall k\in K,\ e\in E_k,\ |e| = 1. \quad (22)$$

First, conditions (20) imply that

$$u_{ik}^t = 0, \quad \forall k\in K,\ \forall i\in H_1^t\backslash\{e_1(k), e_2(k)\}, \text{ if } |e(k)| = 2, \quad (23)$$

$$u_{ik}^t = 0, \quad \forall k\in K,\ \forall i\in H_1^t\backslash\{e_1(k)\}, \text{ if } |e(k)| = 1. \quad (24)$$

Next, conditions (21) and (22) imply that dual slack variables, associated with optimal-primal variables x_{ek}^t set to 1, must be equal to 0. For each $k \in K$, this condition is

$$\alpha_k^t - u_{e_1(k)k}^t - u_{e_2(k)k}^t = F_{e(k)k}, \quad \text{if } |e(k)| = 2, \quad (25)$$

$$\alpha_k^t - u_{e_1(k)k}^t = F_{e(k)k}, \quad \text{if } |e(k)| = 1. \quad (26)$$

This implies that every feasible solution $(\alpha, u) \in D$ satisfying (23)–(26) is indeed an optimal solution of DS. We thus have characterized the set of optimal solutions of the dual subproblem associated with the optimal-primal solution x^t.

Proposition 2. *Let x^t be an optimal solution of PS_k^t. The set of optimal-dual solutions of* DS^t *associated with x^t can be characterized as* $\mathrm{DO}^t = \{(\alpha, u) \in D\colon (23)\text{–}(26)\ \textit{hold}\}$.

The above result implies that we can construct optimal-dual solutions (α^t, u^t) from the optimal-primal solution x^t in two steps. First, we fix each α_k^t, $u_{e_1(k)k}^t$, and $u_{e_2(k)k}^t$, for each $k \in K$, to a particular *feasible* value with respect to constraints (12)–(13) and conditions (25)–(26), and we fix each u_{ik}^t, such that $i \in H_1^t\backslash\{e_1(k), e_2(k)\}$, to zero. Second, we solve a *reduced* system of inequalities by fixing the variables from the first step in constraints (12) and (13), to obtain an optimal value of the remaining u_{ik} such that $i \in H_0^t$, for each $k \in K$.

In the Online Appendix A, we present a procedure for computing an optimal solution (α^t, u^t) from a subset of DO^t associated with solutions in which $\alpha_k^t = F_{e(k)k}$, $u_{e_1(k)k}^t = 0$, and $u_{e_2(k)k}^t = 0$, for each $k \in K$. An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/. By doing so, we avoid checking the feasibility of these variables with respect to constraints (12)–(13). In §4 we present some theoretical insights that help us select particular values of the α_k^t, $u_{e_1(k)k}^t$, and $u_{e_2(k)k}^t$ variables associated with optimal-dual solutions that could produce stronger optimality cuts.

4. Algorithmic Refinements

We now analyze several ways of improving the convergence and stability of the Benders decomposition algorithm presented in the previous section. We first present a multicut version of the Benders reformulation, which exploits the decomposability of the subproblem. Theoretical aspects concerning stronger, nondominated optimality cuts are then introduced and used to develop an algorithm capable of efficiently generating stronger cuts than those presented in §3. Later, we show how to incorporate some reduction tests into the Benders decomposition algorithm in order to reduce the size of both the master problem and the subproblem, and thus accelerate its convergence. Finally, we present a simple heuristic procedure that can be used to generate an initial set of optimality cuts for the master problem to accelerate the convergence of the algorithm and to improve the efficiency of the reduction tests.

4.1. Multicut Benders Reformulation

It is known that the number of cuts required to obtain an optimal solution of the Benders reformulation will be, in

the worst case, equal to the number of extreme points in D. However, this number can be reduced given that the subproblem is decomposable into $|K|$ independent subproblems (see, e.g., Birge and Louveaux 1988). We could in principle generate optimality cuts associated with extreme points of each dual polyhedron of the $|K|$ subproblems, but Camargo et al. (2008) show that when adding $|K|$ cuts per iteration, the reduction in the number of iterations is not justified by the increased computational effort required for the solution of the relaxed master problems, even for small-size instances.

Instead of adding in a disaggregated way all $|K|$ cuts at each iteration, we can aggregate the information obtained to generate a set of optimality cuts associated with subsets of commodities. In particular, for each node $j \in H$, let $K_j \subset K$ be the subset of commodities whose origin node is j. We can separate the subproblem into $|H|$ independent subproblems, one for each node. Hence, we consider the dual polyhedra of these $|H|$ subproblems and generate cuts from them. Let P_D be the set of extreme points of the dual polyhedron P_{D^j} associated with subproblem i. We thus obtain the following Benders reformulation:

$$\text{minimize} \quad \sum_{i\in H} f_i z_i + \sum_{i\in H} \eta_i,$$

$$\text{subject to} \quad (17),\ (18)$$

$$\eta_j \geqslant \sum_{k\in K_j} \alpha_k^t - \sum_{i\in H}\sum_{k\in K_j} u_{ik}^t z_i \qquad \forall\, j \in H,\ \forall\,(\alpha, u) \in P_{D^j}. \quad (27)$$

Using this reformulation, only $|H|$ potential optimality cuts will be generated when solving the subproblem, instead of $|K|$ cuts as is the case when considering the complete separability into $|K|$ dual subproblems.

4.2. Pareto-Optimal Cuts

One way to improve the convergence of the Benders algorithm is to construct stronger, undominated cuts, known as *Pareto-optimal* cuts (Magnanti and Wong 1981). We say that the cut generated from the dual solution (α^a, u^a) dominates the cut generated from the dual solution (α^b, u^b) if and only if $\sum_{k\in K} \alpha_k^a - \sum_{i\in H}\sum_{k\in K} u_{ik}^a z_i \geqslant \sum_{k\in K} \alpha_k^b - \sum_{i\in H}\sum_{k\in K} u_{ik}^b z_i$, for all $z \in Z$ with strict inequality for at least one point. A cut is Pareto optimal if no other cut dominates it. Let Q be the polyhedron defined by (17) and $0 \leqslant z_i \leqslant 1$ for all $i \in H$, and let $ri(Q)$ denote the relative interior of Q. To identify a Pareto-optimal cut at iteration t, we must solve the following *Pareto-optimal subproblem* (PO^t):

$$(\text{PO}^t) \quad \text{maximize} \quad \sum_{k\in K} \alpha_k - \sum_{i\in H}\sum_{k\in K} z_i^0 u_{ik},$$

$$\text{subject to} \quad (12)\text{–}(13),$$

$$\alpha_k - \sum_{i\in H} z_i^t u_{ik} = F_{e(k)k} \quad \forall\, k \in K, \quad (28)$$

where $z^0 \in ri(Q)$ and, as before, $F_{e(k)k}$ is the optimal solution value of subproblem k. Constraints (28) ensure that the optimal solution of PO^t is chosen from the set of optimal solutions of DS^t. Note that PO^t can also be separated into $|K|$ independent subproblems (PO_k^t), one for each $k \in K$. We thus obtain

$$(\text{PO}_k^t) \quad \text{maximize} \quad \alpha_k - \sum_{i\in H} z_i^0 u_{ik}, \quad (29)$$

$$\text{subject to} \quad \alpha_k - \sum_{i\in H} z_i^t u_{ik} = F_{e(k)k} \quad (30)$$

$$\alpha_k - u_{e_1 k} - u_{e_2 k} \leqslant F_{ek} \qquad \forall\, e \in E_k,\ |e| = 2 \quad (31)$$

$$\alpha_k - u_{e_1 k} \leqslant F_{ek} \quad \forall\, e \in E_k,\ |e| = 1 \quad (32)$$

$$u_{ik} \geqslant 0 \quad \forall\, i \in H. \quad (33)$$

Because of constraint (30) and of the *fractional* coefficients z_i^0, the primal structure of (29)–(33) cannot be exploited to efficiently obtain an optimal-dual solution, as is the case for the DS. This means that we need to solve $|K|$ linear programs, one for each $k \in K$, to obtain a Pareto-optimal cut. Computational experiments indicate that the generation of Pareto-optimal cuts considerably reduces the number of required iterations to converge. However, the time needed to solve the $|K|$ linear programs is not compensated by the improved convergence of the Benders algorithm, even on small instances. Given that our goal is to solve large instances, we have developed an efficient procedure capable of producing good approximations of the optimal solution of PO_k^t, and thus of generating stronger optimality cuts without requiring the explicit solution of (29)–(33).

Here we present a procedure capable of efficiently producing stronger optimality cuts, which are not necessarily Pareto-optimal, by exploiting the fact that PO_k^t can be expressed as the maximization of a *piecewise-linear* and *concave* function of α_k. In particular, if we fix the value of the α_k variable in (29)–(33), we can write the resulting subproblem as the following implicit function:

$$L(\alpha_k) = \text{maximize} \quad -\sum_{i\in H} z_i^0 u_{ik},$$

$$\text{subject to} \quad \sum_{i\in H} z_i^t u_{ik} = \alpha_k - F_{e(k)k} \quad (34)$$

$$u_{e_1 k} + u_{e_2 k} \geqslant \alpha_k - F_{ek} \qquad \forall\, e \in E_k,\ |e| = 2 \quad (35)$$

$$u_{e_1 k} \geqslant \alpha_k - F_{ek} \quad \forall\, e \in E_k,\ |e| = 1 \quad (36)$$

$$u_{ik} \geqslant 0 \quad \forall\, i \in H. \quad (37)$$

We now can state PO_k^t as

$$\max_{\alpha_k} G(\alpha_k), \quad (38)$$

where $G(\alpha_k) = \alpha_k - L(\alpha_k)$.

PROPOSITION 3. *$G(\alpha_k)$ is a piecewise-linear and concave function of α_k.*

PROOF. Rewriting the right-hand side vector of constraints (34)–(37) as $b + \alpha_k \hat{b}$, where $b = (-F_{e(k)k}, -F_{1k}, \ldots, -F_{|E_k|k})$ and $\hat{b} = (1, 1, \ldots, 1)$, the problem can be viewed as a linear program in which the right-hand side vector is perturbed along the identity vector. From linear programming theory, we know that parametric analysis on the right-hand side vector in a maximization problem always produces a piecewise-linear and concave function (Bazaraa et al. 1990). Therefore, $L(\alpha_k)$ is a piecewise-linear and concave function of α_k. □

By applying parametric analysis over $L(\alpha_k)$, we can determine the ranges of the linear segments and, thus, the break points at which changes of *optimal* bases (with respect to α_k) take place in $G(\alpha_k)$. Moreover, the slope of each linear segment can be computed using the information of its associated optimal basis. We can therefore obtain an optimal solution of PO_k^t as follows. First, set α_k to some initial feasible value and evaluate $L(\alpha_k)$ to obtain an optimal basis associated with a linear segment. Then, perform as many dual-simplex iterations as break points exist before reaching a point at which the slope of $G(\alpha_k)$ is equal to zero. Even though this procedure is more efficient than solving PO_k^t directly by an LP solver, it still requires the solution of an LP problem to generate an initial optimal basis and its update at each break point.

Instead of optimally solving PO_k^t to produce a Pareto-optimal cut, we solve PO_k^t only approximately and still produce strong, but not necessarily undominated, optimality cuts. Our procedure is based on the estimation of the function $G(\alpha_k)$ by using an adaptation of Algorithm 2 presented in Online Appendix A. Using this estimation, we successively evaluate $G(\alpha_k)$ within a given interval $L_k \leqslant \alpha_k \leqslant U_k$ and increase α_k until the estimation of $G(\alpha_k)$ stops increasing, or until $\alpha_k = U_k$. In Online Appendix B, we present the details on how to efficiently evaluate $G(\alpha_k)$ and how to construct an interval in which the optimal value of α_k is contained.

4.3. Elimination Tests

The efficiency of the Benders decomposition algorithm can be improved by reducing the size of the original model. By doing so, both the master problem and the subproblem can be solved more efficiently. Moreover, the convergence of the algorithm can also benefit from the solution space reduction. In §2 we have presented several optimal UHLPMA properties that can help reduce the size of the model prior to the solution process. Nevertheless, the number of variables and constraints remains very high in large-scale instances.

The size of the model can be further reduced by exploiting the information obtained during the inner iterations of the Benders algorithm. In this section, we develop two different reduction tests capable of eliminating variables that are known not to appear in an optimal solution. Reduction tests have been successfully applied for other HLPs in the context of Lagrangean relaxation (Contreras et al. 2009, 2011). To the best of our knowledge, the idea of using reduction tests within a Benders decomposition algorithm is new.

The first reduction test uses bounds on the optimal solution value to check whether a node may appear in an optimal solution. It exploits the primal information generated during the inner iterations of the Benders algorithm to obtain an estimation of the location and transportation costs associated with feasible solutions containing a hub located at a given node. Using this estimation, we can sometimes determine that the node will not be chosen as a hub. Let MP_{LP}^t denote the linear relaxation of MP^t, $v(\text{MP}_{\text{LP}}^t)$ its optimal solution value, and rc_i the reduced cost associated with variable z_i. The following result provides a reduction test for closing a hub node.

PROPOSITION 4. *Let* UB *be an upper bound on the optimal value of* MP. *If z_i is a nonbasic variable in the optimal solution to MP_{LP}^t and $v(\text{MP}_{\text{LP}}^t) + rc_i > \text{UB}$, then $z_i = 0$ in any optimal solution.*

PROOF. The result follows from the fact that $v(\text{MP}_{\text{LP}}^t) + rc_i$ is a lower bound on the objective function value if a hub is located at node i. Therefore, if $v(\text{MP}_{\text{LP}}^t) + rc_i > \text{UB}$, then $z_i = 0$ in any optimal solution. □

After applying this test, H is updated by removing the eliminated nodes from it. The corresponding node and edge variables are also eliminated from the model.

The second reduction test uses a stronger lower bound that allows checking whether any node in a set of candidate hub nodes $Q \subset H$ may appear in an optimal solution. By solving a slightly modified MP^t, we can obtain an estimation of the total cost associated with feasible solutions containing at least one hub located at a node contained in Q. Using this estimation, we can determine whether $z_i = 0$ for all $i \in Q$ in every optimal solution. We define the following modified master problem:

$$
\begin{aligned}
(\text{MP}^t(Q)) \quad \text{minimize} \quad & \sum_{i \in H} f_i z_i + \sum_{i \in H} \eta_i \\
\text{subject to} \quad & \eta_i \geqslant \sum_{k \in K_i} \alpha_k^t - \sum_{i \in H} \sum_{k \in K_i} u_{ik}^t z_i \\
& \qquad \forall\, i \in H,\ (\alpha, u) \in P_{D^i}^t \\
& \sum_{i \in Q} z_i \geqslant 1 \\
& z_i \in \{0, 1\} \quad \forall\, i \in H.
\end{aligned}
$$

The following result provides the reduction test for closing a set of hub nodes.

PROPOSITION 5. *Let* UB *be an upper bound on the optimal solution value of* MP. *If $v(\text{MP}^t(Q)) > \text{UB}$, then $z_i = 0$ for each $i \in Q$ in any optimal solution.*

Proof. The result follows from the fact that $v(\mathrm{MP}^t(Q))$ is a lower bound on the objective function value if a hub is located at some node $i \in Q$. Therefore, if $v(\mathrm{MP}^t(Q)) > \mathrm{UB}$, then $z_i = 0$ for each $i \in Q$ in any optimal solution. □

For a particular set $Q \subset H$, the previous test requires the solution of an integer linear program. Therefore, we must carefully choose a candidate set Q containing the largest possible number of nodes, while yielding a lower bound strong enough to close the hub nodes. In particular, we want to exclude nodes associated with good feasible solutions of $\mathrm{MP}^t(Q)$ having an objective function value inferior to the upper bound. If we generate a set Q failing the test, we must remove elements from Q so that the resulting set improves the lower bound and passes the test.

The efficiency of the previous test also relies on the quality of the approximation of MP^t. Thus, we should apply the test once we have constructed a sufficiently good approximation of MP. At the beginning of the Benders algorithm we set $Q = H$. Then, at iteration t of the algorithm we discard from Q the set of open hub nodes from the optimal solution of MP^t (i.e., Q is updated to $Q \backslash \{i \in Q\colon z_i^t = 1\}$) as well as the nodes that may have been eliminated through the first test.

When we perform the second elimination test with $\mathrm{MP}^t(Q)$ and it fails, we eliminate from Q the set of open hub nodes from an optimal solution, denoted by $z^t(Q)$, i.e., Q is updated to $Q \backslash \{i\colon z_i^t(Q) = 1\}$. We also eliminate from Q the nodes having small reduced costs $\hat{c}_i$ associated with the LP relaxation, denoted by $\mathrm{MP}^t_{\mathrm{LP}}(Q)$, of $\mathrm{MP}^t(Q)$. In particular, we eliminate from Q nodes such that $\hat{c}_i < \nu \times c_{\max}$, where $c_{\max}$ is the maximum reduced cost associated with nonbasic variables, and ν is a control parameter such that $0 < \nu < 1$. These previous nodes are eliminated from Q only when the gap between the upper bound and the optimal solution value of $\mathrm{MP}^t_{\mathrm{LP}}(Q)$ exceeds a threshold ζ. The proposed procedure is summarized in Algorithm 5 of Online Appendix C.

4.4. A Heuristic Procedure for the UHLPMA

In our Benders reformulation, we know that any vector $z \in Z$ such that $\sum_{i \in H} z_i \geqslant 1$ is a feasible solution for the MP and thus has at least one optimality cut associated with it. We can apply a heuristic to produce a diverse set of feasible solutions, yielding optimality cuts that are incorporated at the beginning of the algorithm. In fact, it is known that the use of an initial approximation of the Benders reformulation polyhedron has a major impact on the required number of iterations (see, e.g., Geoffrion and Graves 1974, Cordeau et al. 2000). The heuristic procedure can also yield good upper bounds that improve the effectiveness of the reduction tests.

We present a simple yet effective heuristic procedure capable of generating high-quality solutions, and diverse solutions, which may provide useful optimality cuts. The proposed heuristic is composed of two phases: an *estimation* phase and an *intensification* phase. The estimation phase is an iterative procedure that constructs a set of initial feasible solutions that are used to construct an interval on the estimated number of open-hub facilities in an optimal solution. The intensification phase is an iterative procedure that generates feasible solutions containing sets of open hubs whose cardinality lies in the interval obtained in the previous phase. Within each phase, we use a common constructive procedure that randomly constructs a feasible solution with a given number of open-hub facilities and improves it by means of a local search procedure. In Online Appendix D, we provide a detailed explanation of the constructive procedure as well as of the overall heuristic.

5. Computational Experiments

We now present the results of extensive computational experiments performed to assess the behavior of our algorithm. In the first part of the computational experiments, we focus on a comparison of different versions of the Benders decomposition algorithm to evaluate the impact of each of the proposed algorithmic features. The second part of the experiments is mainly devoted to a comparison between our exact method and several exact algorithms reported in the literature. In the third part of the experiments, we test the robustness and limitations of our method on large-scale instances involving up to 500 nodes. All algorithms were coded in C and run on a Dell Studio PC with an Intel Core 2 Quad processor Q8200 running at 2.33 GHz and 8 GB of RAM under a Linux environment. The master problems of all versions of the algorithm were solved using the callable library of CPLEX 10.1.

We have used the well-known Australian Post (AP) set of instances to perform the first two parts of the computational experiments. This data set is the most commonly used in the hub location literature. It consists of the Euclidean distances c_{ij} between 200 cities in Australia, of a computer code to reduce the size of the set by grouping cities, and of the values of W_k representing postal flows between pairs of cities. Each instance has a strictly positive flow between every pair of nodes. Therefore, the number of commodities is given by $|K| = |H|^2$. From this set of instances, we have selected those with $|H| = 25$, 50, 75, 100, 125, 150, 175, and 200 and with set-up costs of the type loose (L) (see Contreras et al. 2011, for details). We have varied the required number of open-hub nodes in an optimal solution by increasing the distances of a particular instance as $d_{ij} = TC \times c_{ij}$ for each pair $(i, j) \in H \times H$, where TC is a scaling parameter for the transportation costs. For each instance size we have generated nine different instances corresponding to different combinations of values for the interhub discount factor $\tau \in \{0.2, 0.5, 0.8\}$ and the transportation cost scaling factor $TC \in \{2, 5, 10\}$. Finally, we have considered symmetric transportation costs, i.e., $\chi = \delta = 1$.

In preliminary experiments, we have used the AP instances to set the values of the parameters of the algorithm. The following values were used in all our tests: $\psi = 0.5$, $\gamma = 0.5$, $\kappa = 10$, $\nu = 0.25$, $\zeta = 0.002$, $r_{\max} = 10$, and $z_i^0 = 0.1$ for each $i \in H$. In the first two parts of the experiments, the Benders decomposition algorithm terminated when one of the following criteria was met: (i) the optimality gap between the upper and lower bounds was below a threshold value ϵ, i.e., $|ub - lb|/ub < \epsilon$, (ii) the maximum number of iterations $Iter_{\max}$ was reached; or (iii) the maximum time limit $Time_{\max}$ was reached. We set the parameter values as $\epsilon = 10^{-6}$, $Iter_{\max} = 1{,}000$, and $Time_{\max} = 7{,}200$ seconds.

5.1. Analysis of Algorithmic Refinements

The aim of the first part of the computational experiments is to analyze the effectiveness of each of the algorithmic refinements proposed in §4. For presentation purposes, we only include summarized results of all experiments. The interested reader is referred to Online Appendix F for the detailed results.

We first focus on analyzing the benefits of using the multicut Benders reformulation over the standard Benders reformulation. We have implemented two different versions of Algorithm 1. The first one, called 1-*cut*, uses the reformulation (16)–(18) in which only one optimality cut is added per iteration. The second one, called $|H|$-*cut*, uses the stronger reformulation (17), (18), (27) in which $|H|$ optimality cuts are added per iteration. Both algorithms use Algorithm 2 (Online Appendix A) to generate the optimality cuts at each iteration. The results of the comparison are summarized in Table 1. The first column gives the number of nodes associated with each group of instances. The next two columns under the heading *Optimal Found* give the number of optimal solutions found for 1-*cut* and $|H|$-*cut*, respectively. The next two columns under the heading *Average Time* (*sec*) give the average CPU time in seconds needed to obtain an optimal solution of the problem by using 1-*cut* and $|H|$-*cut*, respectively. This average is computed over the instances that could be solved within the time limit. The last two columns under the heading *Average Iterations* provide the required number of iterations for each of the algorithms to converge. Table 1 shows that both algorithms 1-*cut* and $|H|$-*cut* are able to solve most instances within two hours. However, the strong multicut reformulation is able to solve 71 out of the 72 considered instances, whereas the standard Benders reformulation can solve only 65. The columns *Average time* (*sec*) indicate that $|H|$-*cut* requires on average much less computation time than 1-*cut*. Moreover, as can be seen in the *Average iterations* columns, the convergence of the Benders algorithm is greatly improved by using $|H|$-*cut*. The number of required iterations to converge is reduced by a factor of 10 on average. Given that algorithm $|H|$-*cut* clearly outperforms 1-*cut*, we only consider the multicut Benders reformulation in the remainder of the computational experiments.

Table 1. Comparison of Benders reformulations.

	Optimal found		Average time (sec.)		Average iterations	
$\|H\|$	1-*cut*	$\|H\|$-*cuts*	1-*cut*	$\|H\|$-*cuts*	1-*cut*	$\|H\|$-*cuts*
25	9/9	9/9	8.43	0.67	65.00	9.78
50	9/9	9/9	78.37	4.28	98.11	11.67
75	9/9	9/9	158.37	7.20	87.78	11.11
100	8/9	9/9	1,727.41	54.27	171.11	15.67
125	8/9	9/9	950.24	68.64	119.22	13.56
150	8/9	9/9	1,040.05	166.83	134.11	15.22
175	7/9	9/9	1,885.27	325.66	112.89	12.67
200	7/9	8/9	2,069.38	1,340.87	118.67	18.56
Average	65/72	71/72	989.69	246.05	113.36	13.53

We next focus on analyzing the effectiveness of generating stronger, possibly undominated, optimality cuts. In particular, we have implemented three different versions of Algorithm 1. The first version, referred to as NC, uses the optimality cuts obtained from Algorithm 2 (Online Appendix A). The second version, referred to as POC, uses the Pareto-optimal cuts obtained when solving PO^t by using the dual-simplex algorithm of CPLEX 10.1. The third version, referred to as SC, uses the strong optimality cuts obtained from Algorithm 4 (Online Appendix B). The results of the comparison between these algorithms are summarized in Table 2. The three columns under the heading *Optimal found* give the number of optimal solutions found for each of the considered algorithms. The next columns provide computing times and iterations counts for each version.

The results of Table 2 confirm the efficiency of generating stronger optimality cuts. Both the POC and SC algorithms are able to obtain the optimal solution of all considered instances within two hours of computation time. However, the larger CPU time needed to solve the PO^t problems using POC does not compensate for the improvements in convergence, even for the small-size instances. As can be seen in the *Average time* (*sec*) columns, SC is considerably more efficient than NC and POC. Even though SC generates optimality cuts that are not necessarily Pareto optimal, these seem to be stronger than those used in NC. The *Average iterations* columns also confirm that the convergence of the Benders algorithm can be improved by using SC, but this version is slightly worse than POC. Given that algorithm SC clearly outperforms NC and POC, we only consider the generation of optimality cuts with Algorithm 4 in the rest of the experiments.

In Online Appendix E we provide the computational results related to the analysis of the performance of the heuristic described in §4.4, as well as the analysis on the effect of incorporating the elimination tests of §4.3 into the Benders decomposition algorithm.

Table 2. Comparison of optimality cuts.

	Optimal found			Average time (sec.)			Average iterations		
$\|H\|$	NC	POC	SC	NC	POC	SC	NC	POC	SC
25	9/9	9/9	9/9	0.67	1.13	0.50	9.78	6.78	7.89
50	9/9	9/9	9/9	4.28	9.05	2.10	11.67	7.67	8.00
75	9/9	9/9	9/9	7.20	32.75	5.91	11.11	7.00	7.89
100	9/9	9/9	9/9	54.27	126.78	21.28	15.67	8.22	9.56
125	9/9	9/9	9/9	68.64	366.86	56.72	13.56	9.44	10.67
150	9/9	9/9	9/9	166.83	766.42	123.23	15.22	9.89	11.89
175	9/9	9/9	9/9	325.66	1,130.43	254.76	12.67	8.33	9.56
200	8/9	9/9	9/9	1,340.87	2,275.53	738.05	18.56	10.89	13.78
Average	71/72	72/72	72/72	262.17	588.62	150.32	13.53	8.53	9.90

5.2. Comparison with Alternative Solution Methods

We now present a comparison between our best version of the Benders decomposition algorithm and several exact solution methods previously proposed in the literature. In particular, we compare our exact method with the following five exact algorithms: (i) the Benders decomposition algorithm of Camargo et al. (2008), (ii) the dual adjustment procedure developed by Cánovas et al. (2007), (iii) the relax-and-cut algorithm proposed by Marín (2005), (iv) the solution of a flow-based formulation using CPLEX as described in Boland et al. (2004), and (v) the solution of the strong path-based formulation presented in §2.1, also using CPLEX. To provide a fair comparison, we have run all algorithms on the same computer. The dual adjustment procedure and the relax-and-cut algorithm were obtained from their respective authors, whereas the remaining algorithms were coded by us.

The detailed results of the comparison between the exact methods using the AP data set are provided in Table 3. The first three columns give the number of nodes, the discount factor, and the transportation scale factor. The remaining columns give the CPU time in seconds needed to obtain an optimal solution for each exact algorithm. The *Benders* column provides the results obtained with the best version of our Benders decomposition algorithm. Whenever a solution method cannot optimally solve an instance within two hours of CPU time, we write *time* in the corresponding entry of the table. If an algorithm runs out of memory we then write *memory*.

The results of Table 3 clearly indicate that our exact method outperforms all previously proposed methods. Observe that our algorithm is able to solve all 72 instances whereas the algorithm of Camargo et al. (2008) is only able to optimally solve 52 within two hours of CPU time. The dual adjustment approach by Cánovas et al. (2007) is able to find the optimal solution in 45 out of the 72 instances, and the relax-and-cut approach of Marín (2005) can only solve instances with up to 75 nodes. Similar results are obtained when solving the flow-based formulation of Boland et al. (2004) and the path-based formulation (6)–(10) with CPLEX. This is a clear indication of the limitations of using a general-purpose solver to solve the UHLPMA. In the case of the path-based model, eight GB of memory are not sufficient to load the model into CPLEX when $|H| > 50$. With the flow-based model, larger size instances can be loaded into CPLEX, but their weaker LP bounds do not allow solving instances with more than 75 nodes within two hours of CPU time. It can be seen that our algorithm is always at least one order of magnitude faster than the other exact methods, with the exception of the small instances involving 25 nodes.

5.3. A New Data Set

In the previous experiments, we have shown that the largest size instances of the AP set, containing 200 nodes, can be optimally solved by our algorithm within less than 20 minutes of CPU time. Given that this set contains the largest size instances currently available, we have generated a set of larger instances to test the robustness and limitations of our Benders decomposition algorithm.

At this stage, some comments on the structure of flows in the AP set are in order. We have observed that the amount of flow originating at each node is highly variable in every instance of this set: all instances have a very small number of nodes for which the outgoing flow is much larger than for the other nodes. For example, the 200-nodes instance of the AP set has one node generating 15% of the total flow of the network, another generating 7%, six nodes generating 2%, and the remaining ones each generating less than 1%. This situation seems to make the solution of these instances rather easy because very few nodes have a large impact on the overall cost of the network and thus greatly influence the hub location decisions. As we will show next, instances in which the outgoing flow of each node is within a narrow range are considerably more difficult to solve.

For this reason, we introduce three different sets of instances with diverse structural characteristics in the flow network. In particular, we consider different levels of magnitude for the amount of flow originating at a given node to obtain three different sets of nodes: low-level (LL) nodes, medium-level (ML) nodes, and high-level (HL) nodes. The total outgoing flow of LL, ML and HL nodes lies in the interval $[1, 10]$, $[10, 100]$, and $[100, 1{,}000]$, respectively.

Table 3. Comparison of exact methods with AP instances from 25 to 200 nodes.

			Total time (sec.)					
$\|H\|$	τ	TC	CPLEX	Boland et al. (2004)	Marín (2005)	Cánovas et al. (2007)	Camargo et al. (2008)	Benders
25	0.2	2	1.69	1.37	0.32	0.30	0.24	0.11
	0.2	5	1.36	3.59	0.35	0.19	4.01	0.30
	0.2	10	0.80	2.20	0.25	0.28	73.00	0.85
	0.5	2	1.12	13.78	0.16	0.11	0.20	0.08
	0.5	5	5.29	18.35	0.55	0.29	1.76	0.56
	0.5	10	0.82	5.63	0.29	0.10	10.78	0.44
	0.8	2	0.99	36.22	0.14	0.09	0.19	0.06
	0.8	5	1.04	17.83	0.15	0.06	0.56	0.17
	0.8	10	0.93	9.83	0.12	0.11	2.54	0.25
50	0.2	2	874.46	526.70	56.21	21.36	4.18	0.96
	0.2	5	159.27	644.38	58.24	6.39	20.38	1.76
	0.2	10	57.70	218.54	74.89	12.34	624.41	11.75
	0.5	2	61.62	4,982.49	29.41	5.20	2.65	0.85
	0.5	5	102.06	4,128.70	37.11	5.52	12.04	1.76
	0.5	10	54.58	1,030.34	22.95	6.20	67.16	3.50
	0.8	2	34.34	Time	9.33	2.54	2.33	0.80
	0.8	5	61.31	6,406.17	7.69	1.31	5.35	0.87
	0.8	10	166.38	2,471.61	14.44	7.87	35.04	2.24
75	0.2	2	Memory	6,817.61	4,833.26	104.31	27.43	4.73
	0.2	5	Memory	2,579.95	5,174.35	86.34	73.94	5.80
	0.2	10	Memory	3,045.77	4,916.55	61.12	2,208.01	19.06
	0.5	2	Memory	Time	3,041.93	40.89	21.40	4.78
	0.5	5	Memory	Time	3,287.02	59.71	53.98	5.83
	0.5	10	Memory	Time	2,974.16	24.14	129.12	7.72
	0.8	2	Memory	Time	699.31	35.03	17.02	4.30
	0.8	5	Memory	Time	1,262.82	12.52	50.29	4.43
	0.8	10	Memory	Time	1,438.58	37.82	77.54	7.05
100	0.2	2	Memory	Time	Time	568.43	210.72	15.14
	0.2	5	Memory	Time	Time	1,196.08	1,364.60	28.09
	0.2	10	Memory	Time	Time	1,244.14	Time	58.48
	0.5	2	Memory	Time	Time	182.37	165.75	13.81
	0.5	5	Memory	Time	Time	417.31	787.01	22.61
	0.5	10	Memory	Time	Time	1,762.05	4,586.82	34.77
	0.8	2	Memory	Time	Time	102.84	126.38	13.68
	0.8	5	Memory	Time	Time	155.74	259.70	15.32
	0.8	10	Memory	Time	Time	415.01	1,198.83	18.79
125	0.2	2	Memory	Time	Memory	2,399.08	747.44	47.77
	0.2	5	Memory	Time	Memory	1,584.67	2,406.95	56.34
	0.2	10	Memory	Time	Memory	2,691.29	Time	112.20
	0.5	2	Memory	Time	Memory	625.49	503.83	38.23
	0.5	5	Memory	Time	Memory	836.01	2,150.63	49.51
	0.5	10	Memory	Time	Memory	3,420.10	Time	73.52
	0.8	2	Memory	Time	Memory	179.88	459.49	36.78
	0.8	5	Memory	Time	Memory	547.98	1,061.16	43.55
	0.8	10	Memory	Time	Memory	1,556.03	Time	48.36
150	0.2	2	Memory	Time	Memory	Memory	2,360.82	121.83
	0.2	5	Memory	Time	Memory	Memory	Time	123.90
	0.2	10	Memory	Time	Memory	Memory	2,360.82	121.83
	0.5	2	Memory	Time	Memory	Memory	1,814.04	96.16
	0.5	5	Memory	Time	Memory	Memory	Time	112.44
	0.5	10	Memory	Time	Memory	Memory	Time	135.71
	0.8	2	Memory	Time	Memory	Memory	1,278.42	86.26
	0.8	5	Memory	Time	Memory	Memory	3,816.60	91.66
	0.8	10	Memory	Time	Memory	Memory	6,638.50	105.07
175	0.2	2	Memory	Memory	Memory	Memory	2,062.87	237.55
	0.2	5	Memory	Memory	Memory	Memory	Time	256.08
	0.2	10	Memory	Memory	Memory	Memory	Time	525.67
	0.5	2	Memory	Memory	Memory	Memory	1,776.36	196.25
	0.5	5	Memory	Memory	Memory	Memory	6,575.61	206.45

Table 3. (Continued).

			Total time (sec.)					
$\|H\|$	τ	TC	CPLEX	Boland et al. (2004)	Marín (2005)	Cánovas et al. (2007)	Camargo et al. (2008)	Benders
175	0.5	10	Memory	Memory	Memory	Memory	Time	263.70
	0.8	2	Memory	Memory	Memory	Memory	1,425.14	176.31
	0.8	5	Memory	Memory	Memory	Memory	3,621.65	184.16
	0.8	10	Memory	Memory	Memory	Memory	Time	197.27
200	0.2	2	Memory	Memory	Memory	Memory	Time	483.91
	0.2	5	Memory	Memory	Memory	Memory	Time	485.47
	0.2	10	Memory	Memory	Memory	Memory	Time	1,271.34
	0.5	2	Memory	Memory	Memory	Memory	Time	393.36
	0.5	5	Memory	Memory	Memory	Memory	Time	394.65
	0.5	10	Memory	Memory	Memory	Memory	Time	750.23
	0.8	2	Memory	Memory	Memory	Memory	Time	338.44
	0.8	5	Memory	Memory	Memory	Memory	Time	361.97
	0.8	10	Memory	Memory	Memory	Memory	Time	383.49

Using these nodes, we generate three different classes of instances. In the first set of instances, called *Set* I, the number of HL, ML, and LL nodes is 2%, 38%, and 60% of the total number of nodes, respectively. In the second set, called *Set* II, we construct an instance in such a way that the number of HL, ML, and LL nodes is 30%, 35%, and 35% of the total number of nodes, respectively. Finally, in the third set, called *Set* III, the number of HL, ML, and LL nodes is 0%, 1%, and 99% of the total number of nodes, respectively. In *Set* I we generate instances with $|H| = 50$, 100, 150, 200, 250, 300, 350, 400, 450, and 500. In *Set* II and *Set* III, we generate instances with $|H| = 50$, 100, 150, and 200. For each value of n in each set, we randomly generate the (x, y)-coordinates of the nodes from a continuous uniform distribution in $[0, 1{,}000] \times [0, 1{,}000]$ and define the distance between pairs of nodes as the Euclidean distance. We generate the fixed costs for the hub facilities as $f_i = \theta \times \text{AD}$, where $\theta \sim U[0.3, 0.8]$ and $\text{AD} = \sum_{k \in K} W_k$. Finally, for each basic instance we generate nine instances corresponding to different combinations of values for the interhub discount factor $\tau \in \{0.2, 0.5, 0.8\}$ and the transportation costs scale factor $TC \in \{2, 5, 10\}$. Therefore, *Set* I contains a total of 90 instances, whereas sets *Set* II and *Set* III contain 36 instances each.

In these final computational experiments, we further analyze the performance of the algorithmic refinements, especially the heuristic procedure and the elimination tests. To this end, we consider two different versions of Algorithm 1. The first version, referred to as $B1$, uses the multicut reformulation and the strong optimality cuts obtained from Algorithm 4 (Online Appendix B). However, it does not include the initial cuts nor the elimination tests. The second version, referred to as $B2$, uses the multicut reformulation, the strong cuts, an initial cut associated with the best upper bound found by Algorithm 7 (Online Appendix D), and the two elimination tests. Because of the increase in instance size, we have extended the CPU time limit to one day, i.e., $Time_{\max} = 86{,}400$ seconds.

Computational results are summarized in Tables 4, 5, and 6. The columns *Optimal found* give the number of optimal solutions found by the heuristic, $B1$ and $B2$. The columns *Average % gap* provide the average percent deviation between the best upper and lower bounds, for the heuristic, B1 and B2 when the optimal solution cannot be found within the given time limit. That is, $\%\ gap = 100(\text{UB}_T - \text{LB}_T)/(\text{UB}_T)$, where UB_T and LB_T are the upper and lower bounds, respectively, obtained with $T = B1, B2$. The columns *Average time* (*sec*) provide the average CPU time in seconds needed to obtain an upper bound in the case of the heuristic, and an optimal solution of the problem by using $B1$ and $B2$, respectively. The columns *Average iterations* give the average number of iterations for $B1$ and $B2$. The column *% Closed hubs* gives the average percent of hubs that were closed by the reduction tests in $B2$.

Table 4 shows that both $B1$ and $B2$ algorithms are able to obtain an optimal solution in all instances except one. The relative gap in the remaining instance is 0.67% for $B1$ and 0.59% for $B2$. The heuristic reaches an optimal solution 48 times out of 54. Moreover, the percent deviation in the instances in which the optimal solution could not be found never exceeds 0.8%, and the total average deviation is 0.04%. Algorithm $B2$ is clearly faster than $B1$ on large-scale instances. On 250-nodes and 300-nodes instances, the average CPU time is reduced by half when using the reduction tests and the heuristic procedure. Also, from the *Average iterations* columns we observe that the convergence of the Benders algorithm can be improved by incorporating these features. Moreover, column *% Closed hubs* indicates that a considerable number of candidate hub nodes can be eliminated by using the elimination tests. The percent of closed hubs ranges from 46% to 98%, with an average of 79.62%.

Similar observations can be drawn from Table 5 for *Set* II instances. We observe that these instances are more difficult than those of *Set* I, and the largest instances solved contain only 200 nodes. One possible explanation for this behavior

Table 4. Summary results of 54 instances of *Set* I with $|H| = 50$, 100, 150, 200, 250, and 300.

	Optimal found			Average % gap			Average time (sec.)			Average iterations		
$\|H\|$	Heur	$B1$	$B2$	Heur	$B1$	$B2$	Heur	$B1$	$B2$	$B1$	$B2$	% Closed hubs
50	7/9	9/9	9/9	0.08	0.00	0.00	0.62	1.69	2.11	9.89	8.00	70.89
100	9/9	9/9	9/9	0.00	0.00	0.00	4.52	10.21	12.67	9.33	7.89	79.11
150	9/9	9/9	9/9	0.00	0.00	0.00	21.92	80.44	72.41	14.11	11.33	84.37
200	8/9	9/9	9/9	0.03	0.00	0.00	39.80	321.43	236.49	14.89	11.00	86.06
250	8/9	9/9	9/9	0.00	0.00	0.00	114.58	4,976.90	2,597.97	24.56	17.33	82.89
300	7/9	8/9	8/9	0.12	0.07	0.07	169.05	15,380.02	8,832.61	35.67	30.33	74.41
Average	48/54	53/54	53/54	0.04	0.01	0.01	58.41	3,461.78	1,959.04	18.07	14.31	79.62

Table 5. Summary results of 36 instances of *Set* II with $|H| = 50$, 100, 150, and 200.

	Optimal found			Average % gap			Average time (sec.)			Average iterations		
$\|H\|$	Heur	$B1$	$B2$	Heur	$B1$	$B2$	Heur	$B1$	$B2$	$B1$	$B2$	% Closed hubs
50	9/9	9/9	9/9	0.00	0.00	0.00	0.86	3.00	3.81	8.11	7.22	69.33
100	8/9	9/9	9/9	0.06	0.00	0.00	8.20	27.23	21.61	14.44	11.00	74.56
150	7/9	9/9	9/9	0.09	0.00	0.00	24.80	1,729.73	601.38	23.89	18.22	78.22
200	6/9	7/9	7/9	0.09	0.08	0.06	71.98	1,880.77	861.39	37.78	33.89	71.78
Average	30/36	34/36	34/36	0.06	0.02	0.02	26.46	910.18	372.05	21.06	17.58	73.47

is that the instances in *Set* II no longer have the peculiarity that very few nodes generate a large proportion of the total flow of the network, and thus the decision of where to locate the hubs becomes much more difficult.

Table 6 shows that $B2$ is still superior to $B1$ and that the instances of *Set* III are the most difficult of the test bed. This translates into a smaller percentage of closed hubs and into much longer CPU times.

To better analyze the limit of our algorithm, we have run a final series of computational experiments using the 36 instances of *Set* I with $|H| = 350$, 400, 450, and 500. Given that algorithm $B2$ has proven to be the best version of our Benders decomposition algorithm, these experiments were performed only with this variant. The results of these experiments are summarized in Table 7. They confirm the efficiency and robustness of our algorithm on very large-scale instances. We have proved optimality of 26 out of the 36 considered instances. For the remaining instances, the relative duality gap is below 1%, with a maximum of 1.5% in one instance. The heuristic was able to obtain the optimal or best-known solution in 25 cases out of 36, and the relative deviation for the remaining instances never exceeds 0.7%, except for one instance with 2.56%. From column *% Closed hub* we note that the elimination tests can again

Table 6. Summary results of 36 instances of *Set* III with $|H| = 50$, 100, 150, and 200.

	Optimal found			Average % gap			Average time (sec.)			Average iterations		
$\|H\|$	Heur	$B1$	$B2$	Heur	$B1$	$B2$	Heur	$B1$	$B2$	$B1$	$B2$	% Closed hubs
50	8/9	9/9	9/9	0.06	0.00	0.00	0.85	8.70	8.22	10.89	10.89	61.56
100	7/9	9/9	9/9	0.04	0.00	0.00	8.51	56.60	42.73	16.11	12.00	61.33
150	7/9	9/9	9/9	0.01	0.00	0.00	30.08	5,373.33	1,226.42	34.89	23.44	63.56
200	7/9	7/9	8/9	0.14	0.09	0.02	70.00	5,912.76	2,727.85	38.56	34.67	63.67
Average	29/36	34/36	35/36	0.06	0.02	0.00	27.36	2,837.85	1,001.31	25.11	20.25	62.53

Table 7. Summary results of 36 instances of *Set* I with $|H| = 350$, 400, 450, and 500.

	Optimal found		Average % gap		Average time (sec.)		Average iterations	
$\|H\|$	Heur	$B2$	Heur	$B2$	Heur	$B2$	$B2$	% Closed hubs
350	6/9	7/9	0.33	0.27	359.24	32,141.39	30.33	69.11
400	8/9	8/9	0.04	0.03	610.69	41,844.17	31.67	78.89
450	5/9	5/9	0.17	0.23	1,100.41	19,819.68	33.67	75.38
500	6/9	6/9	0.12	0.23	912.02	31,108.20	23.71	64.69
Average	25/36	26/36	0.16	0.19	745.59	31,228.36	29.85	72.02

close a considerable number of candidate hub nodes. The percent of closed hubs ranges from 2% to 98%, with an average of 72.02%.

6. Conclusions

We have presented an exact Benders decomposition algorithm for large-scale instances of the classical uncapacitated hub location problem with multiple assignments. A standard Benders decomposition was enhanced through the incorporation of several algorithmic features such as a multicut reformulation, the generation of stronger optimality cuts, the incorporation of reduction tests, and the use of a heuristic procedure. Extensive computational experiments on a large set of existing and new instances with up to 500 nodes and 250,000 commodities have clearly confirmed the efficiency and robustness of the algorithm. To the best of our knowledge, the new instances are by far the largest and most difficult ever solved for any type of hub location problem.

7. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/.

Acknowledgments

This work was partly funded by the Canadian Natural Sciences and Engineering Research Council under grants 227837-09 and 39682-10. This support is gratefully acknowledged. Thanks are due to two referees for their valuable comments.

References

Alumur, S., B. Y. Kara. 2008. Network hub location problems: The state of the art. *Eur. J. Oper. Res.* **190**(1) 1–21.

Bazaraa, M. S., J. J. Jarvis, H. D. Sherali. 1990. *Linear Programming and Network Flows*, 2nd ed. Wiley, New York.

Benders, J. F. 1962. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik* **4**(1) 238–252.

Birge, J. R., F. V. Louveaux. 1988. A multicut algorithm for two-stage stochastic linear programs. *Eur. J. Oper. Res.* **34**(3) 384–392.

Boland, N., M. Krishnamoorthy, A. T. Ernst, J. Ebery. 2004. Preprocessing and cutting for multiple allocation hub location problems. *Eur. J. Oper. Res.* **155**(3) 638–653.

Bryan, D. L., M. E. O'Kelly. 1999. Hub-and-spoke networks in air transportation: An analytical review. *J. Regional Sci.* **39**(2) 275–295.

Camargo, R. S., G. Miranda Jr., H. P. Luna. 2008. Benders decomposition for the uncapacitated multiple allocation hub location problem. *Comput. Oper. Res.* **35**(4) 1047–1064.

Campbell, J. F. 1994a. A survey of network hub location. *Stud. Locational Anal.* **6**(1) 31–43.

Campbell, J. F. 1994b. Integer programming formulations of discrete hub location problems. *Eur. J. Oper. Res.* **72**(2) 387–405.

Campbell, J. F. 1996. Hub location and the *p*-hub median problem. *Oper. Res.* **44**(6) 923–935.

Campbell, J. F., A. T. Ernst, M. Krishnamoorthy. 2002. Hub location problems. Z. Drezner, H. W. Hamacher, eds. *Facility Location. Applications and Theory*. Springer, Heidelberg, Germany, 373–408.

Cánovas, L., S. Garcia, A. Marín. 2007. Solving the uncapacitated multiple allocation hub location problem by means of a dual-ascent technique. *Eur. J. Oper. Res.* **179**(3) 990–1007.

Contreras, I., J. A. Díaz, E. Fernández. 2009. Lagrangean relaxation for the capacitated hub location problem with single assignment. *Oper. Res. Spectrum* **31**(3) 483–505.

Contreras, I., J. A. Díaz, E. Fernández. 2011. Branch and price for large-scale capacitated hub location problems with single assignment. *INFORMS J. Comput.* **23**(1) 41–55.

Cordeau, J.-F., F. Soumis, J. Desrosiers. 2000. A Benders decomposition approach for the locomotive and car assignment problem. *Transportation Sci.* **34**(2) 133–149.

Ernst, A. T., M. Krishnamoorthy. 1998. An exact solution approach based on shortest-paths for *p*-hub median problems. *INFORMS J. Comput.* **10**(2) 149–162.

Geoffrion, A. M., G. W. Graves. 1974. Multicommodity distribution system design by Benders decomposition. *Management Sci.* **20**(5) 822–844.

Hamacher, H. W., M. Labbé, S. Nickel, T. Sonneborn. 2004. Adapting polyhedral properties from facility to hub location problems. *Discrete Appl. Math.* **145**(1) 104–116.

Klincewicz, J. G. 1996. A dual algorithm for the uncapacitated hub location problem. *Location Sci.* **4**(3) 173–184.

Klincewicz, J. G. 1998. Hub location in backbone/tributary network design: A review. *Location Sci.* **6**(3) 307–335.

Magnanti, T. L., R. T. Wong. 1981. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Oper. Res.* **29**(3) 464–484.

Marín, A. 2005. Uncapacitated Euclidean hub location: Strengthened formulation, new facets and a relax-and-cut algorithm. *J. Global Optim.* **33**(3) 393–422.

Marín, A., L. Canovas, M. Landete. 2006. New formulations for the uncapacitated multiple allocation hub location problem. *Eur. J. Oper. Res.* **172**(1) 274–292.

Mayer, G., B. Wagner. 2002. HubLocator: An exact solution method for the multiple allocation hub location problem. *Comput. Oper. Res.* **29**(6) 715–739.

O'Kelly, M. E. 1986. The location of interacting hub facilities. *Transportation Sci.* **20**(2) 92–106.

O'Kelly, M. E., H. J. Miller. 1994. The hub network design problem: A review and synthesis. *J. Transport Geography* **2**(1) 31–40.